Preface

This notes constitute an (incomplete!) course script for the first 2 Sessions at the *JAE School* held at ICMAT Madrid, in June 2018.

Maria Isabel Gonzalez Vasco

June 2018

Contents

Session 1

Introduction to Cryptology

On a nutshell:

- **What is Cryptology?**
- **Classical and modern cryptology.**
- **Mathematics and Cryptology: interactions.**

1.1 Getting Started

Definition 1.1.1 *Cryptology:* science devoted to the study, development and critic of techniques for storing, transmitting, modifying and generating information with certain security guarantees.

- *Cryptography:* Constructive side.
- *Cryptanalysis:* Destructive side.

Furthermore, depending on whether we assume or not that the participants involved in the cryptographic task share a high-entropy secret, we speak of:

- *(Public-key/Asymmetric/Modern) Cryptography*: no a priori shared secret;
- *Secret-key/Symmetric/Classical Cryptography*: a priori shared secret assumed.

1.2 Cryptography: Goals and Tools

According to the *Handbook of Applied Cryptology*, most security requirements we can think of are summarized in four:

- *Privacy or Confidentiality:* keeping information secret from all but those who are authorized to see it. This is achieved through different means, ranging from physical protection to mathematical algorithms to render date unintelligible.
- *Data Integrity:* ensuring information has not been altered by unauthorized or unknown means. A key point towards data integrity is being able to detect data manipulation.
- *Authentication:* achieving corroboration of the identity of an entity, or, of the authorship/origin of a certain piece of data.
- *Non-repudiation:* preventing entities from denying previous commitments or actions.

Different cryptographic schemes or protocols allow as to achieve the above goals for concrete application scenarios.

A cryptographic scheme is typically specified by describing:

- *Goal*: what we want to achieve by this construction.
- *Involved Entities*: set of participants which will use the scheme, which may have different abilities and privileges. The communication model is also relevant at this point.
- *Involved Algorithms*: set of algorithms that are executed by these participants.

Furthermore, if we are to give a formal argumentation of the security level achieved by the cryptographic scheme, we should specify the

- *Security Model:* facts we must assume (if any) to base the security of the scheme upon.
  - *Information Theoretical*: no computational assumption is made (to limit the adversarial capabilities).
  - *Computational/Cryptographical*: computational assumptions are made, typically limiting the adversarial capabilities.
- *Adversarial Model:* types of adversaries we are considering (their a priori knowledge, computing power, interaction with the legitimate participants, etc.). At least two types of adversaries are to be considered:
  - *Passive:* only capable of reading information from an insecure channel;
  - *Active:* able to insert, delay, alter or delete information on an insecure channel.

All security models are build upon the so-called *Kerckhofss Principle:*

"Any cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience."

With these goals in mind, different types of cryptographic schemes are designed. The basic ones are often referred to as *cryptographic primitives*; of which we sketch an (incomplete) schematic listing:

- *Unkeyed primitives:*

  - One-way functions
  - Hash functions

- *Secret-Key primitives:*
  - Encryption schemes
  - Message Authentication Codes
  - Pseudorandomnumber Generators
  - Identification primitives

- *Public-Key Primitives*
  - Encryption schemes
  - Digital Signature schemes
  - Identification primitives

In the sequel, we give a brief introduction to hash functions and one-way functions; other primitives will be further explored in the next sessions.

1.2.1 Unkeyed Primitives

One Way Functions. Informally, a *one-way function* is a function f that can be evaluated efficiently, but for which computing pre-images is computationally hard. This (computational) hardness is exploited as a base for security guarantees of cryptographic constructions. More precisely:

Definition 1.2.1 A map $f : \{0,1\}^* \longmapsto \{0,1\}^*$ is *one-way* if the following conditions hold:

- (Easy to compute) There exists a polynomial-time algorithm $\mathcal{M}_f$ computing f, i.e., on input $(x, 1^n)$, $\mathcal{M}_f$ outputs $f(x)$ in time $poly(n)$, for all $n \in \mathbb{N}, x \in \{0,1\}^n$;
- (Hard to invert) Let $\mathcal{A}$ be any probabilistic polynomial time algorithm, then, there exists a negligible function[1] negl such that:

$$P[A(f(u_n), 1^n) \in f^{-1}(f(u_n))] \leq \mathsf{negl}(n)$$

  where u_n is selected u.a.r. in $\{0,1\}^n$.

The above definition is only considering a single function over an infinite domain and range; however, the way to actually formalize the use of one-way functions is slightly different. Typically, we have a (key generation) algorithm which selects a certain parameter i within a countable set I. This parameter defines a certain function f_i of (possibly finite) domain and range, that is (assumed to be) one-way except with negligible probability over the choices from I. For a detailed formal definition, see [**?**].

Since it is not possible to prove the existence of one-way functions, we assume it is existence based on some very natural computational problems, such as *integer factorization* or *discrete logarithm.*

[1] A function ν is *negligible* if for any constant $c > 0$, $\nu(n) = o(n^{-c})$.

Example 1.2.2 Let us show a family of permutations that is believed to be one-way. Suppose we have a key generation algorithm which, on input a security parameter $n \in \mathbb{N}$ outputs and n-bit prime p along with an integer $g \in \{2, ..., p-1\}$ (of large multiplicative order in $\mathbb{Z}_p^*$). Then define

$$f_{p,g}(x) := g^x \mod p.$$

The above procedure defines a family of one-way permutations, under the assumption that the so-called *discrete logarithm problem* is hard.

To conclude; most important even are so-called *trapdoor* one-way functions, which can be informally defined as one-way functions for which an efficient inversion procedure exists, which requires additional (trapdoor) information. Such a function is, as we will see, the bases of the well known RSA encryption scheme.

Hash Functions. Roughly speaking, hash functions are just functions that map bitstrings of arbitrary length into shorter fixed-length bitstrings. As a result, they are highly "non-injective" maps which, however, are often used for detection of data manipulation; i.e., together with a bitstring x, a hash h of it is provided so that a receiver can later check whether $H(x) = h$. Thus, a desirable goal when designing hash functions is to achieve what is called *collision-resistance*; i.e., that an adversary will not succeed in finding two different values hashing to the same bit-string.

Formally,

Definition 1.2.3 A polynomial time computable function $H : \{0,1\}^* \longmapsto \{0,1\}^*$, is a *hash* function, provided that there exist an integer k such that for any $x \in \{0,1\}^*$ $H(x) \in \{0,1\}^k$.

Furthermore, H is called *collision-resistant* if for any probabilistic polynomial time algorithm $\mathcal{A}$ there exists a negligible function negl such that:

$$P[A(H, 1^n) = (x, x') \in \{0,1\}^n \; s.t. \; H(x) = H(x')] \leq \mathsf{negl}(n).$$

Again, the standard way of dealing with hash functions is referring to *hash function families,* which, informally are defined through an index family I that essentially refers to the output length.

1.3 Mathematics of Cryptology

From what we have seen so far it already seems clear that mathematics play a central role in cryptology, for instance as a potential source of hard problems to base the security of cryptographic primitives upon. Furthermore, many mathematical areas and topics have experienced significant developments ultimately inspired by the desire to break cryptographic constructions, which may even have been designed using seemingly unrelated theories. Some of the main areas of mathematics directly linked with research in cryptology are (Computational) Number Theory, Algebraic Geometry, Discrete Mathematics, Group Theory, etc.

However, there is yet an (even more) important aspect of cryptology in which mathematics are crucial; that of yielding formal models in which cryptographic constructions can be described and proven secure.

One could summarize the basic principles of modern cryptography (see [**?**]) in three:

- *Formulation of Exact Definitions.* Nowadays, defining a a security notion boils down to finding the right mathematical model for a real-world problem.
- *Reliance on Precise Assumptions.* Unfortunately, most modern cryptographic constructions cannot be proven secure without computational assumptions (i.e. in an information theoretical sense); thus, the assumptions taken should be made precise in order to be validated by the cryptographic community.
- *Rigorous Proofs of Security.* Informal argumentations of security have historically resulted on weak cryptographic constructions; there are still cases, however, in which ad-hoc solutions are the best one can hope for. Most proofs in modern cryptography are done by reduction (solving problem X is reduced to breaking the scheme Y).

Recommended Readings:

1. A. Menezes, P. van Oorschot and S. Vanstone. Handbook of Applied Cryptography, CRC Press, 1996. *The first chapter contains a comprehensible introduction to cryptology; the complete book, though rather outdated, is still very useful for getting started in the field.*
2. N. Koblitz. Algebraic Aspects of Cryptography, *Springer Verlag, 1999. Nice introduction to the field for mathematicians, good to have an idea of the early years of modern cryptology.*

Session 2

Public Key Encryption

On a nutshell:

- **Definition of Public Key Encryption Schemes**
- **Proving security**
- **RSA Encryption, ELGamal.**

2.1 Definition

Let us suppose we have an scenario in which n participants $P_1, \ldots, P_n$, which have available one-to-one non-private channels, want to be able to exchange messages with some confidentiality guarantees. These participants can be seen as probabilistic polynomial time Turing Machines.

Furthermore, let us assume we are in the *public key* setting, this means, we can not assume that this players share any value, securely agreed upon and thus not accessible to any eavesdropper. We may however assume there is a public and correct directory, accessible to all participants, in which certain values of interest to all participants can be broadcasted.

Now we can describe the algorithms involved in a *public key encryption scheme – PKE*:

Definition 2.1.1 (PKE) A *public key encryption scheme* $\mathsf{S} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is defined by the three following (polynomial) algorithms:

- $\mathcal{K}$, the *key generation algorithm* is a probabilistic algorithm which on input an encoding of the security parameter $n \in \mathbb{N}$, 1^n, produces a pair $(pk, sk)$of matching public and private keys.
- $\mathcal{E}$ the *encryption algorithm* is a probabilistic algorithm which on input a (bitstring) message m –of length $p(n)$ for some polynomial p– and a public key pk produces a ciphertext c –again, a bitstring of length polynomial in n.

- $\mathcal{D}$ the *decryption algorithm* which given a ciphertext c and a secret key sk, outputs either an error message $\perp$ or the corresponding plaintext m.

On the sequel, we will assume the above algorithms to fulfill the obvious correctness condition, i.e., that

$$\mathcal{D}(\mathcal{E}(pk, m), sk) = m$$

for all possible plaintexts m and all key pairs output by $\mathcal{K}$.

Let us start with a first example, the so called Polly Cracker scheme:

Example 2.1.2 Fellows and Koblitz [**?**] introduced the so-called Polly Cracker scheme in 1994. It may be described as follows:

Key generation: fix a polynomial ring $\mathbb{F}_q[x_1, \ldots, x_n]$ over a finite field $\mathbb{F}_q$, and select an ideal $I \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ along with a point $\xi := (\xi_1, \ldots, \xi_n) \in \mathbb{F}_q^n$ in the variety of I (that is, all polynomials of I have ξ as a zero). A finite basis of I forms the public key, whereas the secret key is the point ξ.

Encryption: construct the ciphertext c corresponding to $\sigma \in \mathbb{F}_q$ by selecting a polynomial $p \in I$ and adding it to the plaintext, i. e., $c := \sigma + p \in \mathbb{F}_q[x_1, \ldots, x_n]$.

Decryption: recover the plaintext σ by evaluating c at ξ.

This example already helps us illustrating what a subtle task it may be to argue the security guarantees of an encryption scheme. For instance, in the above description nothing is said about how to generate the finite basis of I or how to concretely construct the "masking polynomial p; clearly, this choices are crucial for the security of the scheme. At least two questions should come up to mind at this point:

- Is the secret key easy to derive from the public information?
- Is there a way to decrypt without the secret key?

Moreover, the answer to this questions is way far from sufficient for building up a solid security proof for the scheme. For instance, even if computing the secret key from the public information is hard, note that, as pointed out by [**?**] if the attacker sends a "fake" ciphertext of the form $x_i + p$ with $p \in I$ and learns the corresponding plaintext, then he learns the i-th coordinate of the secret key. Doing this n times is sufficient to reveal the complete secret key. Does it seem reasonable to allow this?

2.2 Security Analysis

Of course the answer to the above question is no!, and it clearly reflects we were not asking the right questions about the scheme when aiming at the security proof. Let us thus start by enumerating the properties we want our scheme to fulfill:

- *One-Wayness* (OW): with just public data, an attacker cannot retrieve the complete plaintext corresponding to a given ciphertext.

- *Semantic Security* (IND): with just public data, an attacker cannot retrieve any information about the plaintext corresponding to a given ciphertext.
- *Non-malleability* (NM): given several ciphertext and all public data, an adversary should not be able to produce a new valid ciphertext which underlying plaintext is meaningfully related to the plaintexts behind the ciphertexts he was given.

On the other hand, besides specifying the computational power of the adversaries considered, we should moreover make precise assumptions on further types of adversarial capabilities. These are typically formalized as access to "oracles which make certain computations (maybe involving the secret keys) and can be queried by the adversary. This helps us depicting a taxonomy of possible attacks:

- *Chosen Plaintext Attack*(CPA): minimal resources; only public keys accessible;
- *Validity Checking Attack* (VCA): granted access to an oracle which on input a ciphertext and a public key outputs a correctness bit.
- *Chosen Ciphertext Attack*(CCA): granted access to a decryption oracle with minimal limitations (only refuses the challenge ciphertext as input).

Putting together the adversarial goal and capabilities, the standard security notions are build, ranging from OW-CPA (the weakest, in which we require that an adversary, seeing the public key, is not able to decrypt complete ciphertexts) to IND-CCA (the most demanding, where an adversary won't even be able to distinguish which of two plaintexts of his choice, corresponds to the challenge ciphertext, having at hand a CCA oracle.

The following figure summarizes the main relations proven among the most relevant security notions.

$$\begin{array}{ccc} NM-CPA & \longleftarrow & NM-CCA \\ \downarrow & & \Updownarrow \\ IND-CPA & \underset{\not\longrightarrow}{\longleftarrow} & IND-CCA \\ \downarrow\not\uparrow & & \downarrow\not\uparrow \\ OW-CPA & \underset{\not\longrightarrow}{\longleftarrow} & OW-CCA \end{array}$$

2.3 Examples

We now provide informal description of (possibly the most) relevant examples of public key encryption schemes:

Example 2.3.1 RSA scheme was proposed by Rivest, Shamir and Adleman back in 1978 [**?**].

Key generation: On input a security parameter $n \in \mathbb{N}$, select two n-bit primes p and q. Set $N := pq$ and select $e \in \{1, .., \phi(N)\}$ so that $g.c.d.(e, \phi(N)) = 1$. The pair (N, e) constitutes the public key. Compute the secret key $d \in \{1, .., \phi(N)\}$ so that $de \equiv 1 \mod \phi(N)$.

Encryption: given $m \in \{1, \ldots, N\}$ construct the ciphertext $c := m^e \mod N$.

Decryption: recover the plaintext m as $c^d \mod N$.

Proposition 2.3.2 (Informal) *RSA encryption, as described above, is OW-CPA assuming factoring induces a one-way function.*

Example 2.3.3 ElGamal encryption scheme was proposed in 1979 [**?**].

Key generation: On in put a security parameter $n \in N$, generate a cyclic group G of prime order q, with $\log_2 q \sim n$. Select an integer $a \in \{1, \ldots, q-1\}$ and set $y := g^a$. The public key is the triplet (g, q, y), and the secret key is a.

Encryption: given $m \in G$, choose u.a.r. an integer $b \in \{1, \ldots, q-1\}$ and set $c_1 := g^b$, and $c_2 := my^b$.

Decryption: recover the plaintext m as $c_2(c_1^a)^{-1}$.

Proposition 2.3.4 (Informal) *If the* Decisional Diffie-Hellman *problem in G is hard, then, ElGamal encryption, as described above, is IND-CCA.*

Recommended Readings:

1. D. Pointcheval. Provable Security for Public Key Schemes, Chapter of the book Contemporary Cryptology, Birkhauser, 2005. *This chapter contains a comprehensive summary of proof techniques for encryption and digital signature schemes. It is available in the web page of the author.*

2. D. Hofheinz. Public key encryption from a mathematical perspective, *Tutorial from the International School on Mathematical Cryptology held in Barcelona in September 2008. Available on the author's homepage. A very nice, easy to read and complete overview on provable security for encryption schemes.*

Session 3

Public Key Encryption II - IND-CCA Constructions

On a nutshell:

- **The Random Oracle Model**
- **The Bellare Rogaway Construction**
- **The Cramer Shoup Encryption Scheme**

3.1 The Random Oracle Model

Motivated by the hardness of deriving IND-CCA schemes which moreover would be useful in practise (efficient!), people started to introduce several non-standard hypotheses on the adversary. The idea behind these idealizations was mainly to consider only "generic attacks, in the sense that the adversary shall not gain anything from the actual implementation of some involved elements, like hash functions or symmetric block ciphers.

The so called *random oracle model* (ROM) was the first of such models to be introduced [**?**], and is by now widely accepted by the cryptographic community. Essentially, the idea behind this model is to assume that *perfect* hash functions are available for the design, i.e., functions which behave like random functions and give therefore strong collision-resistance guarantees. More precisely, hash functions are formalized by an oracle which produces a truly random value in the range set for each new query. Moreover, if the same value is asked twice, identical answers are obtained. In this section, we will study in depth the generic construction of an INC-CCA encryption scheme in the ROM given by Bellare and Rogaway in [**?**].

3.1.1 The Generic Construction of Bellare and Rogaway

For every natural number $n \in N$, consider fixed finite set X and a trapdoor one-way permutation f onto X. Assume there are two hash functions

$$\mathcal{G} : X \longmapsto \{0,1\}^n \text{ and } H : \{0,1\}^* \longmapsto \{0,1\}^k,$$

(where at this k is polynomial in n).
We describe an encryption scheme $\mathsf{BR} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ as follows:

Key generation: On in put a security parameter $n \in N$, specifies the values needed for computing f and f^{-1}. The public key is thus allowing to evaluate f and the secret key f^{-1}.

Encryption: given $m \in \{0,1\}^r$, choose u.a.r. a value $r \in X$ and set

$$a := f(r), \quad b := m \oplus \mathcal{G}(r) \text{ and } c := \mathcal{H}(m, r).$$

The ciphertext is thus the bitstring $a \parallel b \parallel c$.

Decryption: Using the private key, recover r as $f^{-1}(a)$ and retrieve m as $b \oplus \mathcal{G}(r)$. Check if the received c equals $\mathcal{H}(m, r)$ and output m if so. Otherwise, output $\perp$.

The following result can be obtained:

Proposition 3.1.1 *Let $\mathcal{A}$ be a CCA-adversary against the IND of the above encryption scheme. Suppose he is, given a challenge ciphertext encrypting one of two messages of his choice (m_0 or m_1), able to guess which is the corresponding plaintext with probability $\frac{1}{2} + \varepsilon(n)$, for making q_d, q_g and q_h queries to the decryption oracle, and the hash functions $\mathcal{H}$ and $\mathcal{G}$, respectively. Then, assuming $\mathcal{G}$ and $\mathcal{H}$ are random oracles, he has probability at least*

$$\frac{\varepsilon(n)}{2} - \frac{2q_d}{2^k} - \frac{q_h}{2^n}$$

of violating the one-wayness of f.

3.2 The Cramer Shoup Encryption Scheme

In 1998 Ronald Cramer and Victor Shoup came up with the first practical construction for an IND-CCA encryption scheme which does not require any of the idealization assumptions introduced above. Namely, it works in the standard model under the following assumptions:

- DDH in the underlying cyclic group
- The availability of appropiate universal one-way hash families (weaker than collision resistant).

For each $n \in \mathbb{N}$, let G be a cyclic group of prime order q and p prime with q dividing $p - 1$ (with $\log_2 p = n$). We fix the following notation:

Notation 3.2.1 Denote by $x = (x_1, x_2), y = (y_1, y_2)$ and $z = (z_1, z_2)$ pairs of integers in $\{0, \ldots, q-1\}$. Let $g = (g_1, g_2)$ and $u = (u_1, u_2)$ denote pairs of elements in G. By g^x we denote the group element $g_1^{x_1} g_2^{x_2}$, and similarly, for an integer r, g^{rx} denotes $g_1^{rx_1} g_2^{rx_2}$.

Moreover, consider $\mathcal{H} : \{0, \ldots, p-1\}^3 \longmapsto \{0, \ldots, q-1\}$ a one-way universal hash function.

The encryption scheme $\mathsf{CS} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ can be described as follows:

Key generation: On in put a security parameter $n \in N$, fix p, q, G $\mathcal{H}$ and g as above. Select u.a.r. $x = (x_1, x_2), y = (y_1, y_2)$ and $z = (z_1, z_2)$ in $\{0, \ldots, q-1\}^2$, and set $c := g^x, d := g^y$ and $e := g^z$. The public key is the (x, y, z), while (c, d, e) constitute the secret key.

Encryption: given $m \in G$, choose u.a.r. a value $r \in \{0, \ldots, q-1\}$ and set

$$u_1 := g_1^r, \quad u_2 := g_2^r w := e^r m \text{ and } v := c^r d^{rh}$$

where $h := \mathcal{H}(u_1, u_2, w)$. The ciphertext is thus the 4-tuple (u_1, u_2, w, v)

Decryption: First of all, compute $h = \mathcal{H}(u_1, u_2, w)$ and check $u^{x+hy} = v$. If the check fails,, output $\perp$. Otherwise, retrieve and output $m := w(u^z)^{-1}$.

Remark 3.2.2 Note that any valid ciphertext (u_1, u_2, w, v) fulfills the condition $log_{g_1} u_1 = log_{g_2} u_2$; note that any 4-tuple not fulfilling this will fail the decryption check with overwhelming probability. The reason can be explained as follows, we may see (x, y) as a vector in a 4-dimensional vector space over $\mathbb{F}_q$. The public key constrain "suitable" (x, y) to a 2-dimensional space. Thus if the check fails, as the equation $v = u^{x+hy}$ imposes another independent linear condition on (x, y), only points on a line in S will satisfy it (namely, the probability of finding such (x, y) is $\frac{1}{q}$).

Recommended Readings:

1. D. Pointcheval. Provable Security for Public Key Schemes, Chapter of the book Contemporary Cryptology, Birkhauser, 2005. *This chapter contains a comprehensive summary of proof techniques for encryption and digital signature schemes. It is available in the web page of the author.*

2. N. Koblitz and A. Menezes. Another look at Provable Security, Journal of Cryptology, *Volume 20, Number 1, January 2007. Rather informal, nice introduction to the topic which makes a –sometimes arguable—critique to the stablished provable security paradigms*